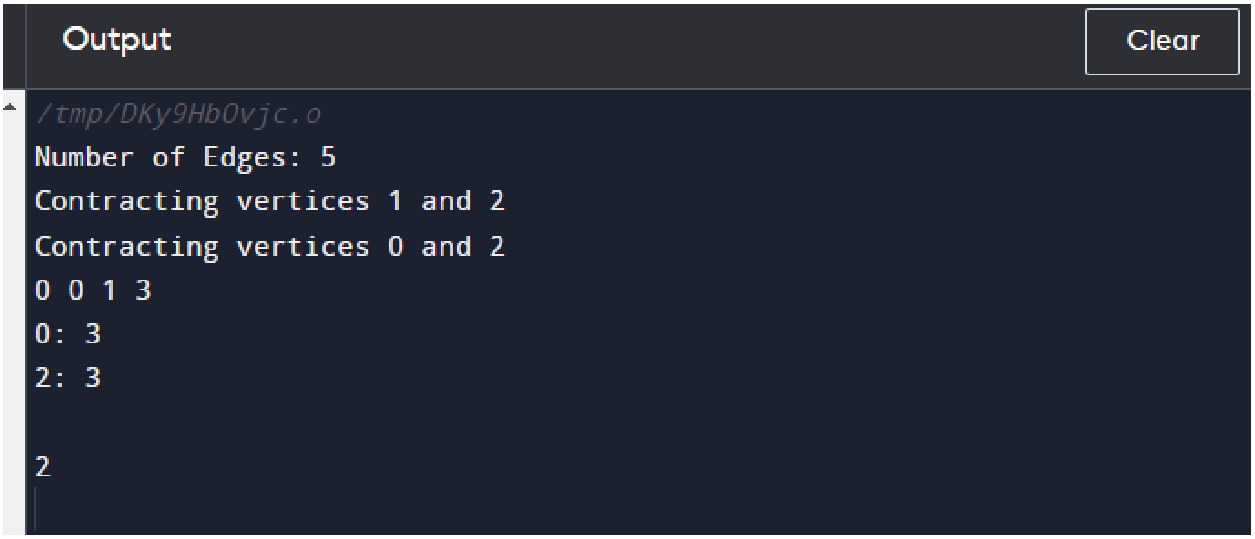
```
#include <iostream>
#define N 4
int G[N][N]={{0,1,1,1},{1,0,1,0},{1,1,0,1},{1,0,1,0}};
int G1[N][N];
int cut=0;
using namespace std;

int edge_count()
{
    int counter=0;
    int i,j;
    for(i=0;i<N;i++)
    {
        for(j=i+1;j<N;j++)
        {
            if(G[i][j]==1)
               counter++;
        }
    }
    return counter;
}
void edge_creation(int edge[][2])
{
    int x=0,i,j;
    for(int i=0;i<N-1;i++)
    {
        for(int j=i+1;j<N;j++)
        {
            if(G[i][j]==1)
            {
                edge[x][0]=i;
                edge[x][1]=j;
                x++;
            }
        }
    }

}
int find(int x,int parent[])
{
        if(x==parent[x])
        {
            return x;
        }
        return parent[x]=find(parent[x],parent);

}
void union_of_edges(int u,int v,int parent[])
{
        u=find(u,parent);
        v=find(v,parent);
        parent[v]=u;
}
int main()
{
    srand(time(NULL));
    int parent[N]={0,1,2,3};
    int m=edge_count();
    cout<<"Number of Edges: "<<m<<endl;
    int i,j;int n=N;
    int edges[m][2];
    edge_creation(edges);


    while(n>2)
    {
        int u,v;
        int e=rand()%m;
        u=edges[e][0];
        v=edges[e][1];
        int set1=find(u,parent);
        int set2=find(v,parent);
        if(set1!=set2)
        {
            cout<<"Contracting vertices "<<u<<" and "<<v<<endl;
            union_of_edges(u,v,parent);
            n--;
        }
    }

   for(i=0;i<N;i++)
    {
      cout<<parent[i]<<" ";
    }
    cout<<endl;
    for(int i=0;i<m;i++)
    {
            int set1=find(edges[i][0],parent);
            int set2=find(edges[i][1],parent);
            if(set1!=set2)
            {
                cout<<edges[i][0]<<": "<<edges[i][1]<<endl;
                cut++;
            }
    }


    cout<<endl<<cut<<endl;
    return 0;
}
```



```
/tmp/DKy9HbOvjc.o
Number of Edges: 5
Contracting vertices 1 and 2
Contracting vertices 0 and 2
0 0 1 3
0: 3
2: 3

2
```